



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD: INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA EN SISTEMAS
CARRERA: SOFTWARE

GUÍA DE LABORATORIO DE APLICACIONES INFORMÁTICAS II

1. DATOS GENERALES:

NOMBRE DEL ESTUDIANTE
Jeferson Vargas

CODIGO DEL ESTUDIANTE
6473

FECHA DE REALIZACIÓN:

2025/07/15

FECHA DE ENTREGA:

2025/07/15

2. OBJETIVO(S):

2.1. GENERAL

- Realizar el análisis de una aplicación (Sistema de facturación) para realizar el proceso de ingeniería inversa.

2.2. ESPECÍFICOS

- Analizar los archivos principales de el sistema para entender su funcionamiento
- Realizar el proceso de ingeniería inversa basándonos en los archivos que identificamos como relevantes

3. METODOLOGÍA

Selección del Aplicativo Informático:

- El primer paso consiste en identificar y seleccionar un aplicativo informático existente sobre el cual se aplicarán las técnicas de ingeniería inversa.
- Se recomienda elegir un software cuya complejidad sea manejable para el alcance de la práctica, pero que a su vez permita la aplicación efectiva de las técnicas.

Aplicación de la Ingeniería Inversa:

- Se deben utilizar técnicas de **análisis estático** (sin ejecutar el código) para examinar la estructura del software, como la revisión del código fuente (si está disponible), el uso de desensambladores (como IDA Pro o Ghidra para binarios) y decompiladores (como JD-GUI para Java o dnSpy para .NET) para obtener representaciones de alto nivel del código.
- Adicionalmente, se puede emplear el **análisis dinámico** (observando el comportamiento del software durante su ejecución) utilizando depuradores (como OllyDbg o GDB) para entender los flujos de ejecución, el uso de memoria y las interacciones con el sistema operativo o la red (mediante herramientas como Wireshark).
- El objetivo es comprender a fondo los componentes del aplicativo, sus interrelaciones, los flujos de datos, la lógica de negocio y la arquitectura inferida.

4. EQUIPOS Y MATERIALES:

- Computador
- Entorno integrado de desarrollo (IDE)
- Aula virtual
- Acceso a internet
- Bibliografía

5. ANALISIS DEL SISTEMA

El sistema analizado, del cual se ha examinado un componente clave, tiene como **propósito principal facilitar la emisión y gestión de comprobantes de pago electrónicos, específicamente facturas, de acuerdo con la normativa de la Superintendencia Nacional de Aduanas y de Administración Tributaria (SUNAT) de Perú**. Su función central es automatizar el proceso de creación de documentos fiscales digitales, su firma electrónica y su envío a la autoridad tributaria, garantizando así la validez y el cumplimiento fiscal de las transacciones comerciales.

En esencia, sirve como un **intermediario programático** entre las operaciones de negocio internas y los exigentes requisitos técnicos y legales de la facturación electrónica peruana, transformando datos comerciales en documentos fiscales válidos y comunicándolos eficientemente con SUNAT.

Funcionalidades Principales

Basándonos en el código y la estructura observada, las funcionalidades principales de este sistema son:

Modelado de Documentos Electrónicos:

- Permite la **estructuración programática de una factura electrónica (Invoice)**, incluyendo todos los campos requeridos por SUNAT como la versión UBL, fechas de emisión y vencimiento, tipo de operación, tipo de documento (01 para factura), serie y número correlativo, moneda (PEN), y campos de observación.
- Capacidad para **definir los detalles de venta (SaleDetail)** de cada ítem de la factura, especificando datos como código de producto, descripción, unidad de medida, cantidad, valores unitarios (con y sin IGV), montos de IGV, y el tipo de afectación al impuesto.
- Soporte para la **inclusión de leyendas (Legend)** en la factura, como la representación textual del monto total.

Gestión de Datos Maestros (Empresa y Cliente):

- A través de la clase auxiliar Util y su componente shared, el sistema abstrae la obtención de los **datos de la empresa emisora** (Razon Social, RUC, Dirección, etc.) y los **datos del cliente receptor** (Tipo de Documento, Número de Documento, Nombre/Razón Social, etc.). Esta centralización sugiere una funcionalidad de gestión de entidades previamente cargadas o configuradas.

Generación y Envío a la Autoridad Tributaria:

- Orquesta la **conexión con los servicios web de la SUNAT** (o un intermediario como beatose.herokuapp.com) mediante la configuración de un servicio de envío (Greenter\See).
- Realiza el **envío automatizado de la factura** al servicio de SUNAT, lo que implica la generación interna del XML bajo el estándar UBL, la aplicación de la firma digital (gestionada por Greenter), y la transmisión segura del documento.

Manejo Básico de Respuestas Fiscales:

- **Verificación del Estado de Envío:** Capacidad para determinar si el documento fue recibido y validado exitosamente por SUNAT.
- **Gestión de Errores:** En caso de fallos en el envío o validación por parte de SUNAT, el sistema captura y muestra el mensaje de error.
- **Procesamiento del CDR:** Si el envío es exitoso, el sistema es capaz de obtener y manejar el Comprobante de Recepción (CDR) emitido por SUNAT, el cual es la constancia oficial de la validez del comprobante.

Persistencia de Documentos (Auditoría):

- Utiliza la clase Util para **guardar los archivos XML** de las facturas generadas y los archivos **ZIP que contienen el CDR** recibido de SUNAT. Esta funcionalidad es crucial para el cumplimiento y la auditoría, permitiendo almacenar una copia local de los documentos fiscales emitidos y sus respuestas.

6. ARQUITECTURA

El análisis de ingeniería inversa del sistema de facturación, basado en el código fuente proporcionado (especialmente el script de emisión de facturas y las referencias a la clase Util), revela una arquitectura pragmática que combina elementos de un diseño en **Capas** con la aplicación parcial del patrón **Modelo-Vista-Controlador (MVC)**, y la utilización de patrones de diseño específicos como el **Singleton**.

Enfoque Arquitectónico Principal: Arquitectura en Capas

El sistema demuestra una clara separación de responsabilidades a través de una arquitectura en capas lógicas:

- **Capa de Orquestación/Controlador:** El script factura-beatose.php se posiciona como una capa de control y orquestación. Su función es recibir la solicitud de emisión de una factura, coordinar la preparación de los datos, interactuar con la lógica de negocio y los servicios de integración, y gestionar la respuesta final. En un sistema más amplio, esta capa podría ser un controlador de una aplicación web o un procesador de un servicio de fondo.
- **Capa de Lógica de Negocio/Dominio:** Esta capa contiene las reglas y el procesamiento central relacionados con la facturación.

- Los **Modelos de Greenter** (Greenter\Model\Sale\Invoice, Greenter\Model\Sale\SaleDetail, Greenter\Model\Sale\Legend) actúan como representaciones de las entidades de dominio (la factura y sus ítems) y encapsulan la estructura de datos conforme a las especificaciones UBL y SUNAT.
 - La **Clase Util** se identifica como un componente clave en esta capa. Aunque su implementación completa no fue proporcionada, su rol inferido abarca lógica auxiliar crítica para el negocio, como la obtención de datos maestros de la empresa y el cliente, la gestión de la configuración del servicio SUNAT, y las operaciones de E/S de archivos para documentos fiscales.
- **Capa de Acceso a Datos (Inferida):** Se infiere la existencia de una capa de acceso a datos que interactúa con la **Clase Util**. Esta capa sería responsable de la persistencia y recuperación de datos maestros (empresa, clientes) de una base de datos subyacente. Adicionalmente, Util gestiona el acceso a datos relacionados con el almacenamiento de los archivos XML de las facturas generadas y los ZIP de los Comprobantes de Recepción (CDR) de SUNAT en el sistema de archivos.
- **Capa de Integración con Servicios Externos:** Esta capa gestiona la comunicación con entidades externas y servicios especializados.
 - La **Librería Greenter** constituye el componente principal de esta capa, encapsulando la complejidad de la comunicación con los servicios web de la SUNAT (firma digital, protocolos de comunicación).
 - El **Endpoint de SUNAT** (o del intermediario beatose.herokuapp.com) representa el servicio externo final con el que el sistema interactúa para la validación y registro de los comprobantes electrónicos.

Patrones de Diseño Aplicados

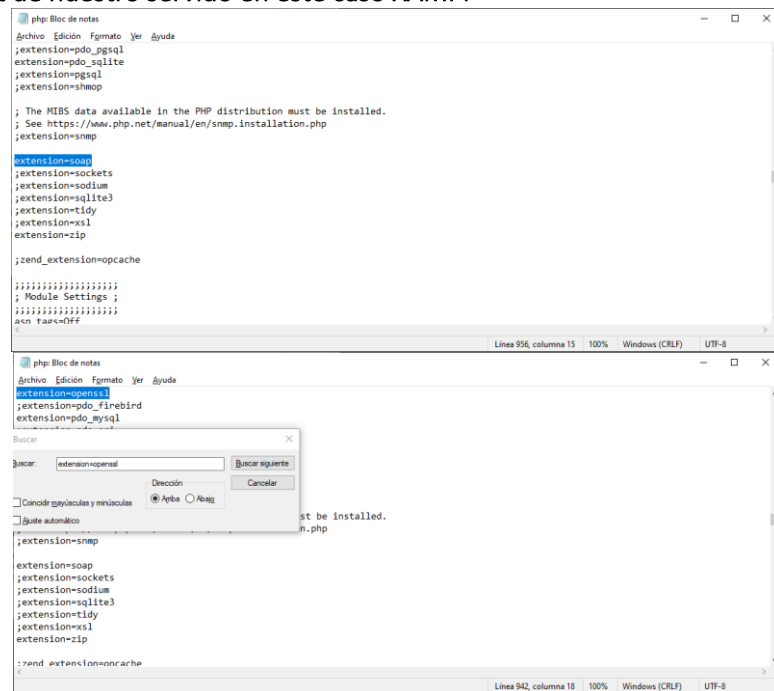
El análisis revela la aplicación de patrones de diseño que contribuyen a la estructura y funcionalidad del sistema:

- **Modelo-Vista-Controlador (MVC) - Aplicación Parcial:** Si bien el sistema no se presenta como un *framework* MVC completo, se observan elementos clave:
 - **Modelos:** Claramente representados por las clases de Greenter\Model (Invoice, SaleDetail, etc.), que encapsulan la información estructurada del documento fiscal.
 - **Controlador:** El script factura-beatose.php asume el rol de un controlador, orquestando el flujo desde la preparación de la factura hasta la gestión de la respuesta del servicio externo. La ausencia de una "Vista" explícita en el fragmento sugiere que la salida (errores, confirmaciones) podría manejarse en la consola o ser integrada en una capa de presentación superior.
- **Singleton:** La clase Util implementa el patrón **Singleton** (Util::getInstance()). Este patrón garantiza que solo exista una única instancia de Util a lo largo de la ejecución del programa. Esta elección de diseño es adecuada para componentes que gestionan recursos compartidos o globales, como configuraciones del sistema, credenciales o conexiones a servicios, asegurando una gestión centralizada y consistente.
- **Fluent Interface (Method Chaining):** El uso extensivo de encadenamiento de métodos (ej., \$invoice->setUblVersion(...)->setFecVencimiento(...)->...) es un patrón que mejora la legibilidad del código al permitir que múltiples operaciones sobre un mismo objeto se realicen de manera concisa y consecutiva.

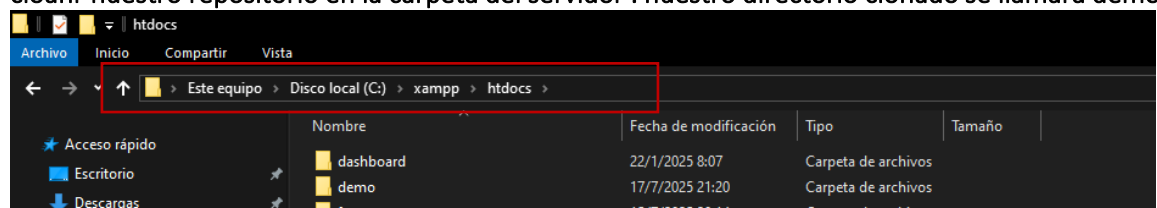
7. Análisis Estático

Como primer paso realizamos la configuración del entorno en el que se puede desplegar el sistema para poder utilizarlo en modo local, basándonos en el requerimiento del código alojado en

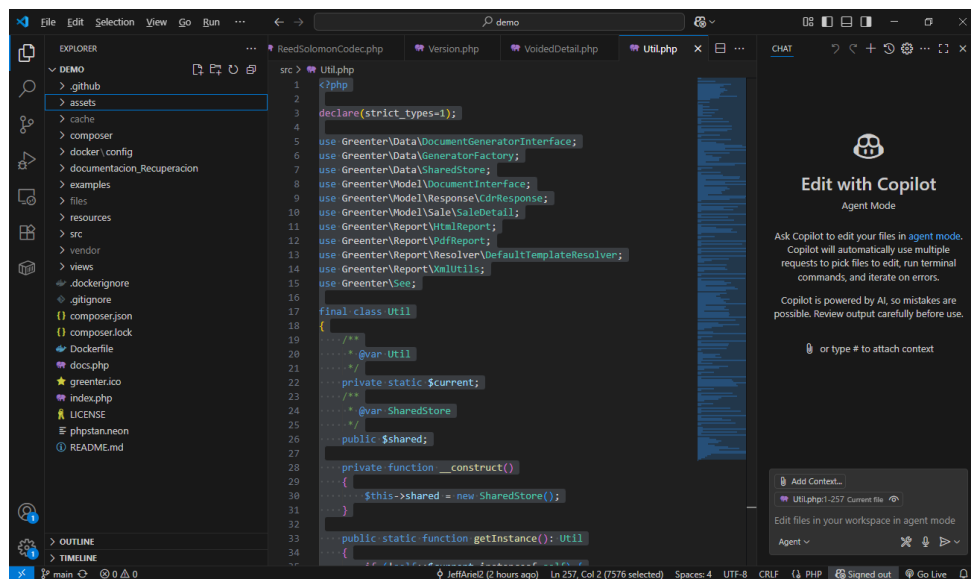
el repositorio de GitHub comprobamos que necesitamos php en sus versiones actualizadas y configuradas con las extensiones soap, y openssl activadas para lo cual se modifican en el archivo php init de nuestro servidor en este caso XAMPP



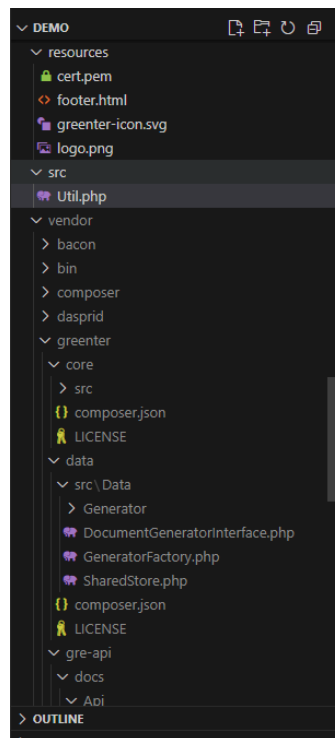
Una vez configurado el servidor de XAAMP en el que uqeremos correr el aplicativo procedemos a cloanr nuestro repositorio en la carpeta del servidor . nuestro directorio clonado se llamara demo



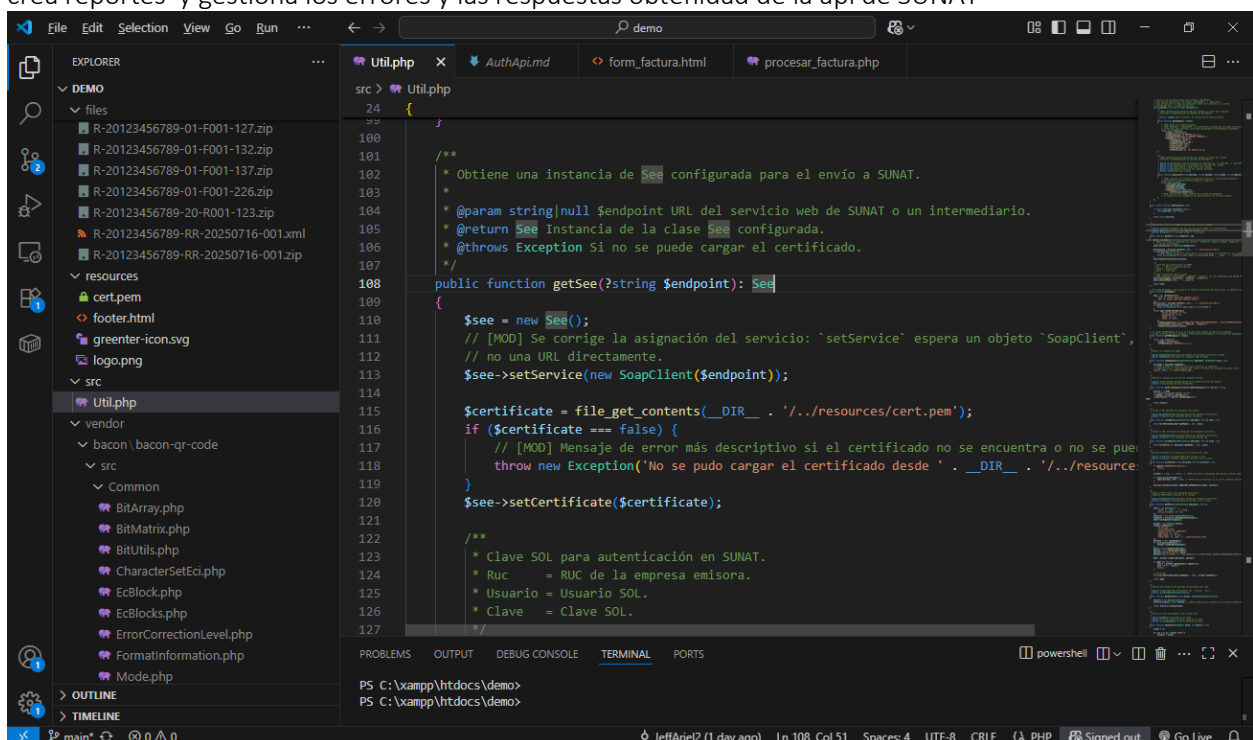
Para el análisis del código fuente y también de emos elegido el IDE Visual studio Code para lo cual copiamos nuestra carpeta a nuestro IDE para poder analizar de esta manera el sistema en un análisis estático.



Observando la estructura del directorio nos podemos fijar que tiene muchos documentos de diferente formato lo cual podemos deducir que el sistema tiene lenguajes de JavaScript, PHP, Html,CSS, entre otros,



Una vez integrado el directorio a nuestro ide porcedemos a hacer una análisis breve de los ficheros de los cuales se puede deducir los ficheros mas importantes par asu funcionamiento como son útil.ph Este archivo se encarga de la gestión global y proporciona una única instancia (con el patron sigleton) para acceder a las configuración y servicios de manera consistente en toda la aplicacionademas centraliza la lógica para obtener y proporcionar los datos de la empresa emisora y los datos dinámico del cliente que proviene de la entrada del usuario, gestiona también al coenxcion con la api de SUNat y con el certificado digital y las credenciales SOL necesarias para al autenticación, también maneja los archivos crea reportes y gestiona los errores y las respuestas obtenidad de la api de SUNAT



también podemos nombrar como archivo importarte el archivo Authapi.md en este archivo se define una el api e cliente que interactúa con el servidor de seguridad de SUNAT, para la generación de e tokens de acceso. Esto es diseñado para obtener un token de autenticación del servicio de seguridad de SUNAT. también presenta URI relativas al api de seguridad de seguridad de sunat esto especifica el endpoint para la autenticación y a la gestión de tokens. AuthApi es la interfaz programática para obtener los tokens de acceso de SUNAT, lo cual es un paso fundamental en el proceso de integración con los nuevos servicios web de facturación electrónica que implementan OAuth2.

The image displays two screenshots of a code editor showing the `AuthApi.md` file. The top screenshot shows the 'Parameters' section, which is a table of API parameters. The bottom screenshot shows the 'Return type' section, which is a PHP code snippet for the `getToken` method.

Top Screenshot: Parameters Table

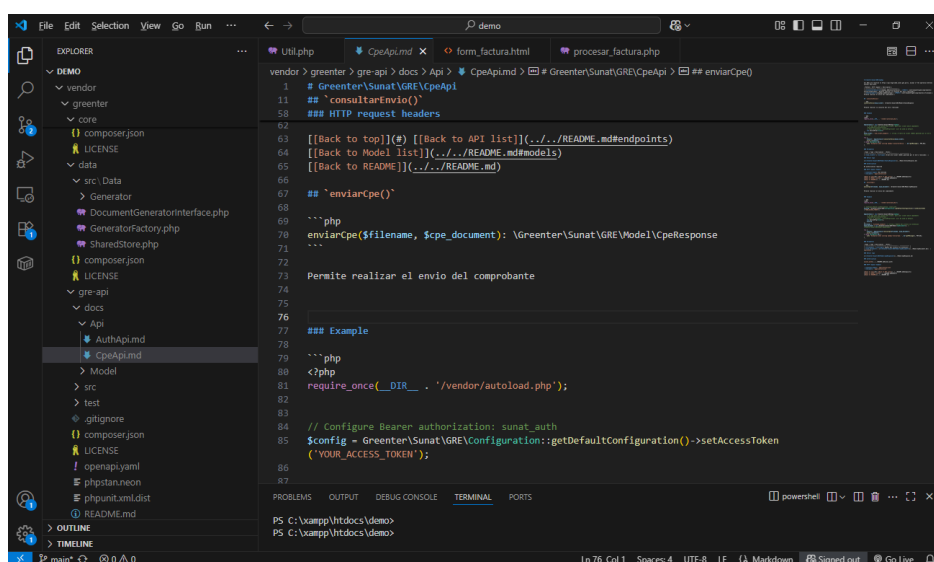
Name	Type	Description	Notes
<code>client_id</code>	<code>string</code>	El client_id generado en menú sol	
<code>grant_type</code>	<code>string</code>	[default to &39;password&39;]	
<code>scope</code>	<code>string</code>	[default to &39;https://api-cpe.sunat.gob.pe&39;]	
<code>client_id2</code>	<code>string</code>	client_id generado en menú sol	
<code>client_secret</code>	<code>string</code>	client_secret generado en menú sol	
<code>username</code>	<code>string</code>	<Numero de RUC> + <Usuario SOL>	
<code>password</code>	<code>string</code>	Contraseña SOL	

Bottom Screenshot: Return type

```

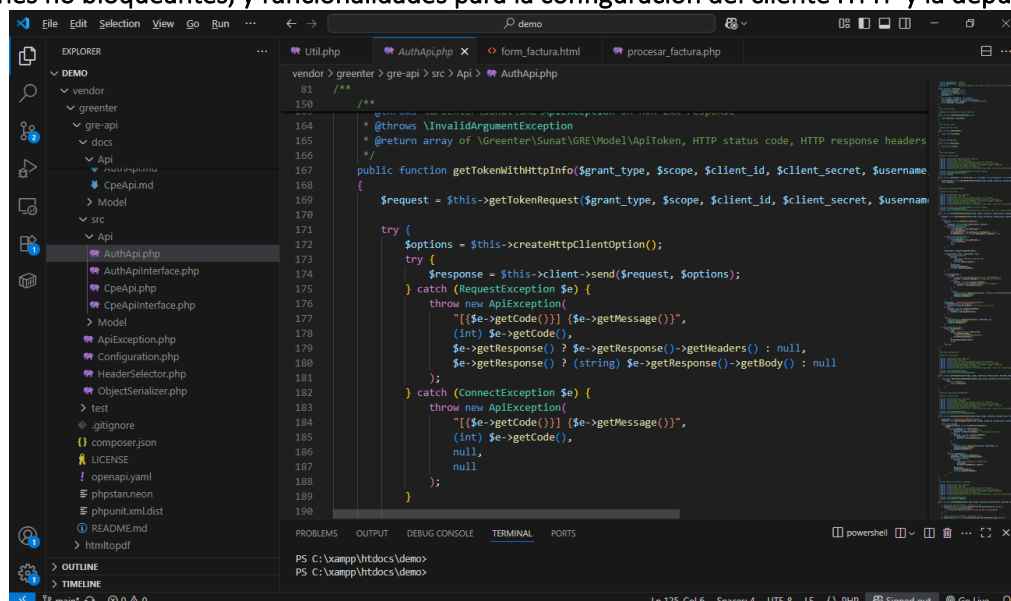
1  # Greenter\Sunat\GRE\AuthApi
10 ## `getToken()`
20 ### Example
37 $client_secret = 'client_secret_example'; // string | client_secret generado en menú sol
38 $username = 'username_example'; // string | <Numero de RUC> + <Usuario SOL>
39 $password = 'password_example'; // string | Contraseña SOL
40
41 try {
42     $result = $apiInstance->getToken($client_id, $grant_type, $scope, $client_id2, $client_secret,
43                                     $username, $password);
44     print_r($result);
45 } catch (Exception $e) {
46     echo 'Exception when calling AuthApi->getToken: ', $e->getMessage(), PHP_EOL;
47 }
48
49 ### Parameters
50
51 | Name | Type | Description | Notes |
52 |-----|-----|-----|-----|
53 | **client_id** | **string** | El client_id generado en menú sol | |
54 | **grant_type** | **string** | [default to &39;password&39;] | |
55 | **scope** | **string** | [default to &39;https://api-cpe.sunat.gob.pe&39;] | |
56 | **client_id2** | **string** | client_id generado en menú sol | |
57 | **client_secret** | **string** | client_secret generado en menú sol | |
58 | **username** | **string** | &lt;Numero de RUC&gt; + &lt;Usuario SOL&gt; | |
59 | **password** | **string** | Contraseña SOL | |
60
61 ### Return type
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2
```

La clase CpeApi.md actúa como la interfaz principal para interactuar con los servicios web de SUNAT en el contexto de los Comprobantes de Pago Electrónicos (CPE) bajo el nuevo esquema de autenticación OAuth2. Su funcionalidad más importante radica en la capacidad de enviar comprobantes electrónicos a la SUNAT a través del método enviarCpe(), el cual espera un objeto CpeDocument que encapsula el XML del comprobante, y requiere un token de acceso OAuth2 previamente obtenido para la autenticación. Adicionalmente, esta API proporciona el método consultarEnvio(), cuya funcionalidad destacada es permitir verificar el estado de un envío ya realizado a SUNAT utilizando el número de ticket (numTicket) recibido en la respuesta del envío inicial. Ambas operaciones utilizan GuzzleHttp\Client para las peticiones HTTP y sus URLs base apuntan a los servicios de SUNAT relacionados con la gestión de comprobantes.



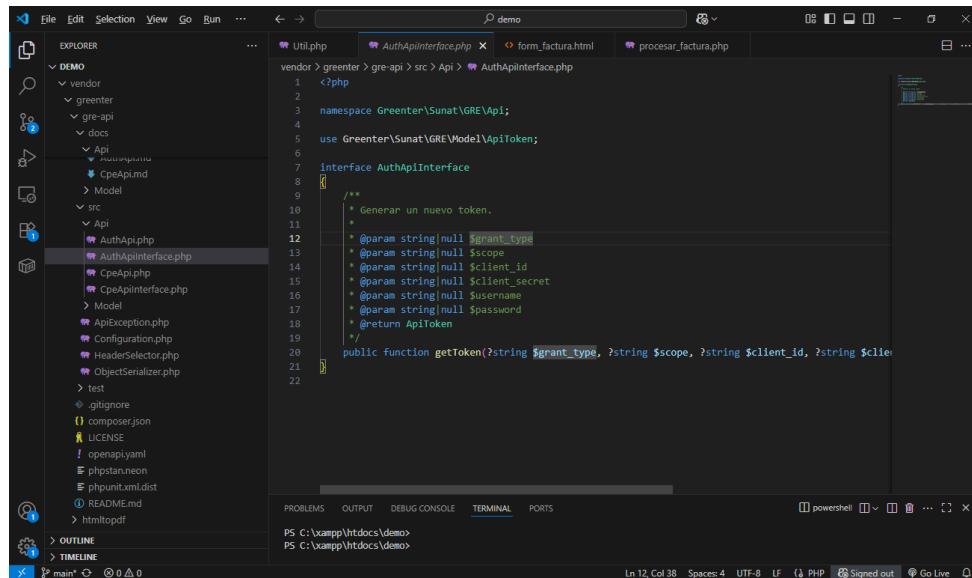
```
vendor > greenter > gre-api > docs > Api > CpeApi.md > # Greenter\Sunat\GRE\CpeApi > ## enviarCpe()
1  # Greenter\Sunat\GRE\CpeApi
11 # #consultarEnvio()
58 ### HTTP request headers
62
63 [[Back to top]](# [[Back to API list]](../README.md#endpoints)
64 [[Back to Model list]](../README.md#models)
65 [[Back to README]](../README.md)
66
67 # # 'enviarCpe()'
68
69 ```php
70 enviarCpe($filename, $cpe_document): \Greenter\Sunat\GRE\Model\CpeResponse
71 ...
72
73 Permite realizar el envío del comprobante
74
75
76
77 ### Example
78
79 ```php
80 <?php
81 require_once( __DIR__ . '/vendor/autoload.php');
82
83
84 // Configure Bearer authorization: sunat_auth
85 $config = Greenter\Sunat\GRE\Configuration::getDefaultConfiguration()->setAccessToken(
86     'YOUR_ACCESS_TOKEN');
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Este código PHP, generado automáticamente por OpenAPI Generator, define la clase AuthApi que funciona como un cliente de API robusto y estandarizado cuya misión principal es obtener tokens de autenticación OAuth2 del servicio de seguridad de SUNAT. La clase facilita esta tarea a través de su método principal getToken(), el cual se encarga de construir y ejecutar la solicitud HTTP (utilizando GuzzleHttp) con las credenciales de la empresa (client_id, client_secret, RUC + Usuario SOL, y Contraseña SOL) y los parámetros de OAuth2 (grant_type, scope), manejando de forma interna la validación de parámetros, la gestión de errores de conexión y de la API, y la deserialización de la respuesta JSON en un objeto ApiToken para su uso posterior. Además, incluye versiones asíncronas (getTokenAsync()) para operaciones no bloqueantes, y funcionalidades para la configuración del cliente HTTP y la depuración.



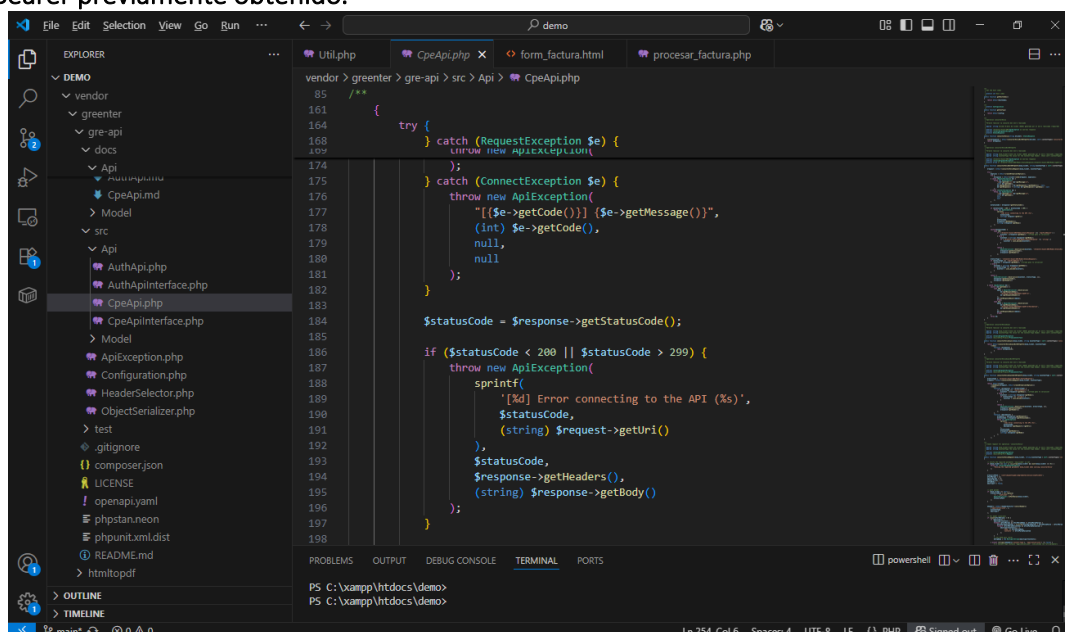
```
vendor > greenter > gre-api > src > Api > AuthApi.php
81 /**
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164 * @throws \InvalidArgumentException
165 * @return array of \Greenter\Sunat\GRE\Model\ApiToken, HTTP status code, HTTP response headers
166 */
167 public function getTokenWithHttpInfo($grant_type, $scope, $client_id, $client_secret, $username)
168 {
169     $request = $this->getTokenRequest($grant_type, $scope, $client_id, $client_secret, $username);
170
171     try {
172         $options = $this->createHttpClientOption();
173         try {
174             $response = $this->client->send($request, $options);
175         } catch (RequestException $e) {
176             throw new ApiException(
177                 "[{$e->getCode()}] {$e->getMessage()}",
178                 (int) $e->getStatusCode(),
179                 $e->getResponse() ? $e->getResponse()->getHeaders() : null,
180                 $e->getResponse() ? (string) $e->getResponse()->getBody() : null
181             );
182         } catch (ConnectException $e) {
183             throw new ApiException(
184                 "[{$e->getCode()}] {$e->getMessage()}",
185                 (int) $e->getStatusCode(),
186                 null,
187                 null
188             );
189         }
190     }
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```


Este código PHP define la interfaz `AuthApiInterface`, que establece un contrato formal para las clases que interactúan con el servicio de autenticación de SUNAT. Su única función es declarar el método `getToken()`, especificando que cualquier clase que implemente esta interfaz debe tener un método público con ese nombre, el cual acepta seis parámetros opcionales de tipo cadena (para el tipo de concesión, el alcance, el ID y secreto del cliente, el nombre de usuario y la contraseña) y debe devolver un objeto de tipo `\Greenter\Sunat\GRE\Model\ApiToken`, asegurando así una estructura consistente para la obtención de tokens de autenticación.



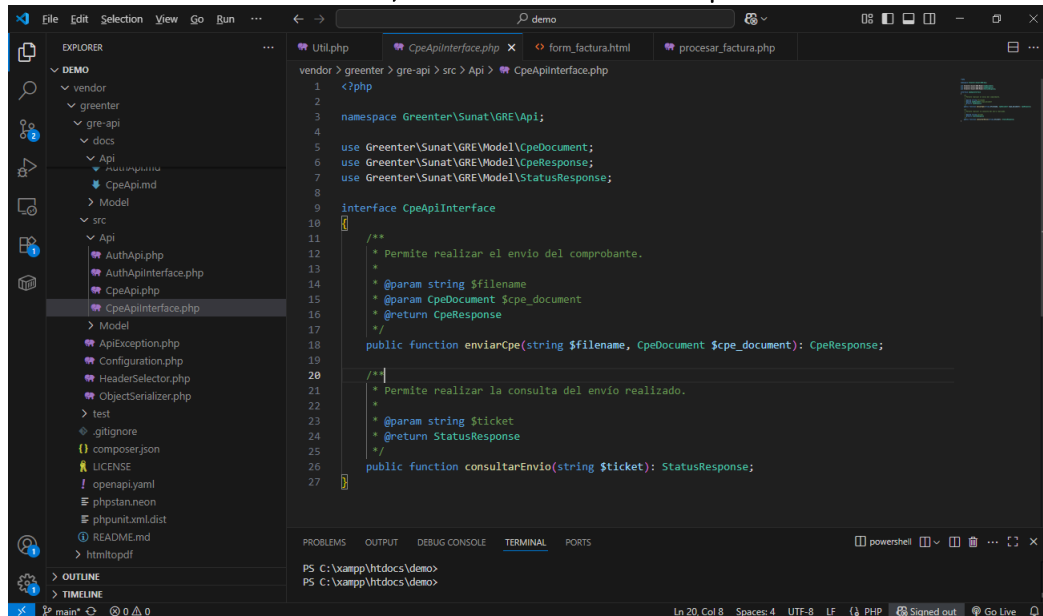
```
1 <?php
2
3 namespace Greenter\Sunat\GRE\Api;
4
5 use Greenter\Sunat\GRE\Model\ApiToken;
6
7 interface AuthApiInterface
8 {
9     /**
10      * Generar un nuevo token.
11      *
12      * @param string|null $grant_type
13      * @param string|null $scope
14      * @param string|null $client_id
15      * @param string|null $client_secret
16      * @param string|null $username
17      * @param string|null $password
18      * @return ApiToken
19      */
20     public function getToken(?string $grant_type, ?string $scope, ?string $client_id, ?string $client_secret, ?string $username, ?string $password);
21 }
22
```

Este código PHP define la clase `CpeApi`, un cliente de API generado automáticamente para interactuar con los servicios de la Plataforma Nueva GRE de SUNAT relacionados con los Comprobantes de Pago Electrónicos (CPE), implementando la interfaz `CpeApiInterface` para garantizar un contrato de métodos bien definido. Sus funcionalidades principales se centran en la gestión de CPEs: el método `enviarCpe()` permite enviar comprobantes electrónicos a SUNAT, requiriendo un filename para la URL y un objeto `CpeDocument` que encapsula el XML del comprobante, mientras que `consultarEnvio()` habilita la consulta del estado de un envío previo mediante el `numTicket` (UUID) generado por SUNAT, con ambos métodos gestionando internamente la construcción de peticiones HTTP con Guzzle, el manejo de respuestas (incluyendo la deserialización a `CpeResponse` o `StatusResponse`) y la robusta gestión de errores y excepciones de comunicación y de la API de SUNAT, además de asegurar la autenticación mediante un token Bearer previamente obtenido.



```
185 /**
186  *
187  * @param string $url
188  * @param CpeDocument $cpeDocument
189  * @return StatusResponse
190  */
191 public function enviarCpe(string $url, CpeDocument $cpeDocument): StatusResponse
192 {
193     try {
194         $response = $this->httpClient->post($url, [
195             'headers' => [
196                 'Content-Type' => 'application/xml',
197                 'Authorization' => 'Bearer ' . $this->getToken(),
198             ],
199             'body' => $cpeDocument->getXml(),
200         ]);
201         $statusCode = $response->getStatusCode();
202
203         if ($statusCode < 200 || $statusCode > 299) {
204             throw new ApiException(
205                 sprintf(
206                     '[%d] Error connecting to the API (%s)',
207                     $statusCode,
208                     (string) $response->getBody()
209                 ),
210                 $statusCode,
211                 $response->getHeaders(),
212                 (string) $response->getBody()
213             );
214         }
215
216         return StatusResponse::from($response->getBody());
217     } catch (RequestException $e) {
218         throw new ApiException(
219             sprintf(
220                 '[%d] Request Exception (%s)',
221                 $e->getStatusCode(),
222                 $e->getMessage()
223             ),
224             $e->getStatusCode(),
225             $e->getHeaders(),
226             (string) $e->getBody()
227         );
228     } catch (ConnectException $e) {
229         throw new ApiException(
230             sprintf(
231                 '[%d] Connect Exception (%s)',
232                 $e->getStatusCode(),
233                 $e->getMessage()
234             ),
235             $e->getStatusCode(),
236             $e->getHeaders(),
237             (string) $e->getBody()
238         );
239     }
240 }
241
```

Este código PHP define la interfaz `CpeApiInterface`, que actúa como un contrato formal para las clases destinadas a interactuar con los servicios de Comprobantes de Pago Electrónicos (CPE) de SUNAT, garantizando que cualquier implementación de esta interfaz ofrezca dos funcionalidades esenciales: el método `enviarCpe()`, que especifica la necesidad de enviar un filename y un objeto `CpeDocument` para el comprobante, retornando un `CpeResponse`, y el método `consultarEnvio()`, que permite verificar el estado de un envío anterior a través de un ticket, devolviendo un `StatusResponse`.



```
1 <?php
2
3 namespace Greenter\Sunat\GRE\Api;
4
5 use Greenter\Sunat\GRE\Model\CpeDocument;
6 use Greenter\Sunat\GRE\Model\CpeResponse;
7 use Greenter\Sunat\GRE\Model>StatusResponse;
8
9 interface CpeApiInterface
10
11 /**
12  * Permite realizar el envío del comprobante.
13  *
14  * @param string $filename
15  * @param CpeDocument $cpe_document
16  * @return CpeResponse
17  */
18 public function enviarCpe(string $filename, CpeDocument $cpe_document): CpeResponse;
19
20 /**
21  * Permite realizar la consulta del envío realizado.
22  *
23  * @param string $ticket
24  * @return StatusResponse
25  */
26 public function consultarEnvio(string $ticket): StatusResponse;
27
```

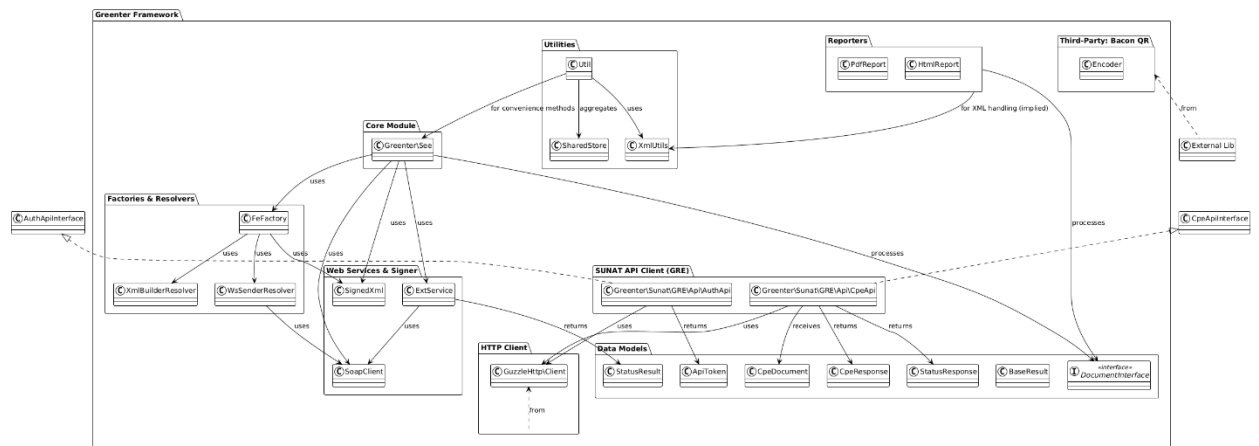
Este código PHP define la clase `See` (Sistema de Emisión del Contribuyente), que actúa como un cliente centralizado para la emisión de documentos electrónicos (CPE) hacia SUNAT. La clase se encarga de orquestar todo el proceso, desde la generación y firma digital del XML de los comprobantes hasta su envío directo a los servicios web de SUNAT y la consulta de su estado. Internamente, utiliza inyección de dependencias y resolutores para configurar dinámicamente el constructor de XML (basado en Twig para plantillas) y el remitente del servicio web (utilizando un `SoapClient` y manejando credenciales SOL o secundarias), mientras gestiona el certificado digital para la firma con `SignedXml`, permitiendo tanto el envío de objetos `DocumentInterface` como de cadenas XML pregeneradas, e incluyendo la capacidad de interpretar el tipo y nombre de archivo a partir del contenido XML para facilitar el proceso de envío.

The screenshot shows a code editor with the following content:

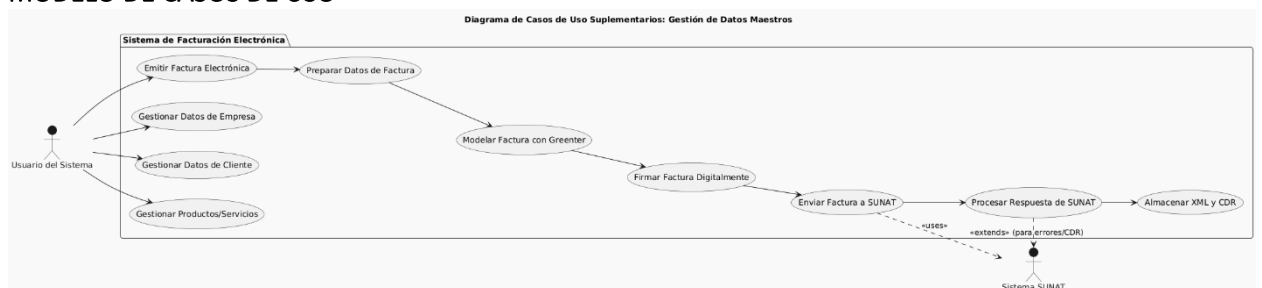
```

vendor > greenter > lite > src > Greenter > Factory > FeFactory.php

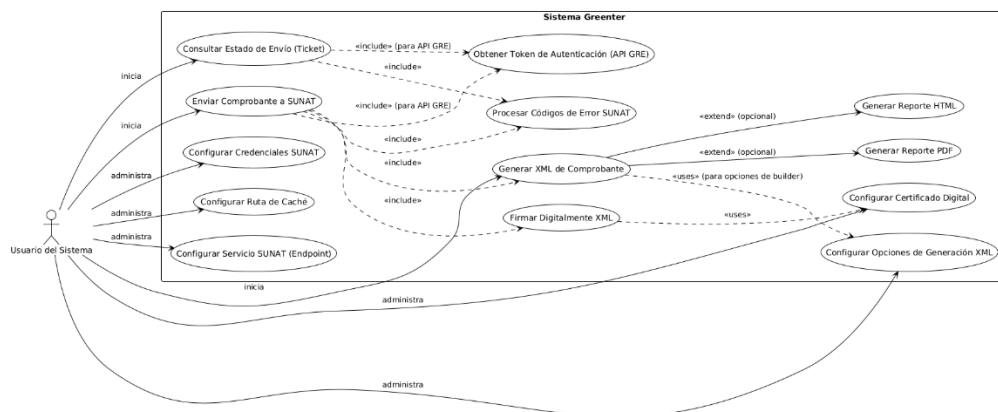
2  /**
3   */
4
5  declare(strict_types=1);
6
7  namespace Greenter\Factory;
8
9  use Greenter\Builder\BuilderInterface;
10 use Greenter\Model\DocumentInterface;
11 use Greenter\Model\Response\BaseResult;
12 use Greenter\Services\SenderInterface;
13 use Greenter\XMLSecLibs\Sunat\SignedXml;
14
15 /**
16  * Class FeFactory.
17  */
18 class FeFactory implements FactoryInterface
19 {
20     /**
21      * @var SignedXml|null
22      */
23     private $signer;
24
25     /**
26      * Sender service.
27      */
28     * @var SenderInterface|null
29
30     /**
31      * Constructor
32      */
33     public function __construct($signer, $sender)
34     {
35         $this->setSigner($signer);
36         $this->setSender($sender);
37     }
38
39     /**
40      * Set signer
41      */
42     public function setSigner($signer)
43     {
44         $this->signer = $signer;
45     }
46
47     /**
48      * Set sender
49      */
50     public function setSender($sender)
51     {
52         $this->sender = $sender;
53     }
54
55     /**
56      * Get signer
57      */
58     public function getSigner()
59     {
60         return $this->signer;
61     }
62
63     /**
64      * Get sender
65      */
66     public function getSender()
67     {
68         return $this->sender;
69     }
70
71     /**
72      * Create invoice
73      */
74     public function createInvoice($data)
75     {
76         // ...
77     }
78
79     /**
80      * Create sale detail
81      */
82     public function createSaleDetail($data)
83     {
84         // ...
85     }
86
87     /**
88      * Create legend
89      */
90     public function createLegend($data)
91     {
92         // ...
93     }
94
95     /**
96      * Create document
97      */
98     public function createDocument($data)
99     {
100        // ...
101    }
102
103    /**
104     * Create response
105     */
106    public function createResponse($data)
107    {
108        // ...
109    }
110
111    /**
112     * Create result
113     */
114    public function createResult($data)
115    {
116        // ...
117    }
118
119    /**
120     * Create error
121     */
122    public function createError($data)
123    {
124        // ...
125    }
126
127    /**
128     * Create exception
129     */
130    public function createException($data)
131    {
132        // ...
133    }
134
135    /**
136     * Create exception
137     */
138    public function createException($data)
139    {
140        // ...
141    }
142
143    /**
144     * Create exception
145     */
146    public function createException($data)
147    {
148        // ...
149    }
150
151    /**
152     * Create exception
153     */
154    public function createException($data)
155    {
156        // ...
157    }
158
159    /**
160     * Create exception
161     */
162    public function createException($data)
163    {
164        // ...
165    }
166
167    /**
168     * Create exception
169     */
170    public function createException($data)
171    {
172        // ...
173    }
174
175    /**
176     * Create exception
177     */
178    public function createException($data)
179    {
180        // ...
181    }
182
183    /**
184     * Create exception
185     */
186    public function createException($data)
187    {
188        // ...
189    }
190
191    /**
192     * Create exception
193     */
194    public function createException($data)
195    {
196        // ...
197    }
198
199    /**
200     * Create exception
201     */
202    public function createException($data)
203    {
204        // ...
205    }
206
207    /**
208     * Create exception
209     */
210    public function createException($data)
211    {
212        // ...
213    }
214
215    /**
216     * Create exception
217     */
218    public function createException($data)
219    {
220        // ...
221    }
222
223    /**
224     * Create exception
225     */
226    public function createException($data)
227    {
228        // ...
229    }
230
231    /**
232     * Create exception
233     */
234    public function createException($data)
235    {
236        // ...
237    }
238
239    /**
240     * Create exception
241     */
242    public function createException($data)
243    {
244        // ...
245    }
246
247    /**
248     * Create exception
249     */
250    public function createException($data)
251    {
252        // ...
253    }
254
255    /**
256     * Create exception
257     */
258    public function createException($data)
259    {
260        // ...
261    }
262
263    /**
264     * Create exception
265     */
266    public function createException($data)
267    {
268        // ...
269    }
270
271    /**
272     * Create exception
273     */
274    public function createException($data)
275    {
276        // ...
277    }
278
279    /**
280     * Create exception
281     */
282    public function createException($data)
283    {
284        // ...
285    }
286
287    /**
288     * Create exception
289     */
290    public function createException($data)
291    {
292        // ...
293    }
294
295    /**
296     * Create exception
297     */
298    public function createException($data)
299    {
300        // ...
301    }
302
303    /**
304     * Create exception
305     */
306    public function createException($data)
307    {
308        // ...
309    }
310
311    /**
312     * Create exception
313     */
314    public function createException($data)
315    {
316        // ...
317    }
318
319    /**
320     * Create exception
321     */
322    public function createException($data)
323    {
324        // ...
325    }
326
327    /**
328     * Create exception
329     */
330    public function createException($data)
331    {
332        // ...
333    }
334
335    /**
336     * Create exception
337     */
338    public function createException($data)
339    {
340        // ...
341    }
342
343    /**
344     * Create exception
345     */
346    public function createException($data)
347    {
348        // ...
349    }
350
351    /**
352     * Create exception
353     */
354    public function createException($data)
355    {
356        // ...
357    }
358
359    /**
360     * Create exception
361     */
362    public function createException($data)
363    {
364        // ...
365    }
366
367    /**
368     * Create exception
369     */
370    public function createException($data)
371    {
372        // ...
373    }
374
375    /**
376     * Create exception
377     */
378    public function createException($data)
379    {
380        // ...
381    }
382
383    /**
384     * Create exception
385     */
386    public function createException($data)
387    {
388        // ...
389    }
390
391    /**
392     * Create exception
393     */
394    public function createException($data)
395    {
396        // ...
397    }
398
399    /**
400     * Create exception
401     */
402    public function createException($data)
403    {
404        // ...
405    }
406
407    /**
408     * Create exception
409     */
410    public function createException($data)
411    {
412        // ...
413    }
414
415    /**
416     * Create exception
417     */
418    public function createException($data)
419    {
420        // ...
421    }
422
423    /**
424     * Create exception
425     */
426    public function createException($data)
427    {
428        // ...
429    }
430
431    /**
432     * Create exception
433     */
434    public function createException($data)
435    {
436        // ...
437    }
438
439    /**
440     * Create exception
441     */
442    public function createException($data)
443    {
444        // ...
445    }
446
447    /**
448     * Create exception
449     */
450    public function createException($data)
451    {
452        // ...
453    }
454
455    /**
456     * Create exception
457     */
458    public function createException($data)
459    {
460        // ...
461    }
462
463    /**
464     * Create exception
465     */
466    public function createException($data)
467    {
468        // ...
469    }
470
471    /**
472     * Create exception
473     */
474    public function createException($data)
475    {
476        // ...
477    }
478
479    /**
480     * Create exception
481     */
482    public function createException($data)
483    {
484        // ...
485    }
486
487    /**
488     * Create exception
489     */
490    public function createException($data)
491    {
492        // ...
493    }
494
495    /**
496     * Create exception
497     */
498    public function createException($data)
499    {
500        // ...
501    }
502
503    /**
504     * Create exception
505     */
506    public function createException($data)
507    {
508        // ...
509    }
510
511    /**
512     * Create exception
513     */
514    public function createException($data)
515    {
516        // ...
517    }
518
519    /**
520     * Create exception
521     */
522    public function createException($data)
523    {
524        // ...
525    }
526
527    /**
528     * Create exception
529     */
530    public function createException($data)
531    {
532        // ...
533    }
534
535    /**
536     * Create exception
537     */
538    public function createException($data)
539    {
540        // ...
541    }
542
543    /**
544     * Create exception
545     */
546    public function createException($data)
547    {
548        // ...
549    }
550
551    /**
552     * Create exception
553     */
554    public function createException($data)
555    {
556        // ...
557    }
558
559    /**
560     * Create exception
561     */
562    public function createException($data)
563    {
564        // ...
565    }
566
567    /**
568     * Create exception
569     */
570    public function createException($data)
571    {
572        // ...
573    }
574
575    /**
576     * Create exception
577     */
578    public function createException($data)
579    {
580        // ...
581    }
582
583    /**
584     * Create exception
585     */
586    public function createException($data)
587    {
588        // ...
589    }
590
591    /**
592     * Create exception
593     */
594    public function createException($data)
595    {
596        // ...
597    }
598
599    /**
600     * Create exception
601     */
602    public function createException($data)
603    {
604        // ...
605    }
606
607    /**
608     * Create exception
609     */
610    public function createException($data)
611    {
612        // ...
613    }
614
615    /**
616     * Create exception
617     */
618    public function createException($data)
619    {
620        // ...
621    }
622
623    /**
624     * Create exception
625     */
626    public function createException($data)
627    {
628        // ...
629    }
630
631    /**
632     * Create exception
633     */
634    public function createException($data)
635    {
636        // ...
637    }
638
639    /**
640     * Create exception
641     */
642    public function createException($data)
643    {
644        // ...
645    }
646
647    /**
648     * Create exception
649     */
650    public function createException($data)
651    {
652        // ...
653    }
654
655    /**
656     * Create exception
657     */
658    public function createException($data)
659    {
660        // ...
661    }
662
663    /**
664     * Create exception
665     */
666    public function createException($data)
667    {
668        // ...
669    }
670
671    /**
672     * Create exception
673     */
674    public function createException($data)
675    {
676        // ...
677    }
678
679    /**
680     * Create exception
681     */
682    public function createException($data)
683    {
684        // ...
685    }
686
687    /**
688     * Create exception
689     */
690    public function createException($data)
691    {
692        // ...
693    }
694
695    /**
696     * Create exception
697     */
698    public function createException($data)
699    {
700        // ...
701    }
702
703    /**
704     * Create exception
705     */
706    public function createException($data)
707    {
708        // ...
709    }
710
711    /**
712     * Create exception
713     */
714    public function createException($data)
715    {
716        // ...
717    }
718
719    /**
720     * Create exception
721     */
722    public function createException($data)
723    {
724        // ...
725    }
726
727    /**
728     * Create exception
729     */
730    public function createException($data)
731    {
732        // ...
733    }
734
735    /**
736     * Create exception
737     */
738    public function createException($data)
739    {
740        // ...
741    }
742
743    /**
744     * Create exception
745     */
746    public function createException($data)
747    {
748        // ...
749    }
750
751    /**
752     * Create exception
753     */
754    public function createException($data)
755    {
756        // ...
757    }
758
759    /**
760     * Create exception
761     */
762    public function createException($data)
763    {
764        // ...
765    }
766
767    /**
768     * Create exception
769     */
770    public function createException($data)
771    {
772        // ...
773    }
774
775    /**
776     * Create exception
777     */
778    public function createException($data)
779    {
780        // ...
781    }
782
783    /**
784     * Create exception
785     */
786    public function createException($data)
787    {
788        // ...
789    }
790
791    /**
792     * Create exception
793     */
794    public function createException($data)
795    {
796        // ...
797    }
798
799    /**
800     * Create exception
801     */
802    public function createException($data)
803    {
804        // ...
805    }
806
807    /**
808     * Create exception
809     */
810    public function createException($data)
811    {
812        // ...
813    }
814
815    /**
816     * Create exception
817     */
818    public function createException($data)
819    {
820        // ...
821    }
822
823    /**
824     * Create exception
825     */
826    public function createException($data)
827    {
828        // ...
829    }
830
831    /**
832     * Create exception
833     */
834    public function createException($data)
835    {
836        // ...
837    }
838
839    /**
840     * Create exception
841     */
842    public function createException($data)
843    {
844        // ...
845    }
846
847    /**
848     * Create exception
849     */
850    public function createException($data)
851    {
852        // ...
853    }
854
855    /**
856     * Create exception
857     */
858    public function createException($data)
859    {
860        // ...
861    }
862
863    /**
864     * Create exception
865     */
866    public function createException($data)
867    {
868        // ...
869    }
870
871    /**
872     * Create exception
873     */
874    public function createException($data)
875    {
876        // ...
877    }
878
879    /**
880     * Create exception
881     */
882    public function createException($data)
883    {
884        // ...
885    }
886
887    /**
888     * Create exception
889     */
890    public function createException($data)
891    {
892        // ...
893    }
894
895    /**
896     * Create exception
897     */
898    public function createException($data)
899    {
900        // ...
901    }
902
903    /**
904     * Create exception
905     */
906    public function createException($data)
907    {
908        // ...
909    }
910
911    /**
912     * Create exception
913     */
914    public function createException($data)
915    {
916        // ...
917    }
918
919    /**
920     * Create exception
921     */
922    public function createException($data)
923    {
924        // ...
925    }
926
927    /**
928     * Create exception
929     */
930    public function createException($data)
931    {
932        // ...
933    }
934
935    /**
936     * Create exception
937     */
938    public function createException($data)
939    {
940        // ...
941    }
942
943    /**
944     * Create exception
945     */
946    public function createException($data)
947    {
948        // ...
949    }
950
951    /**
952     * Create exception
953     */
954    public function createException($data)
955    {
956        // ...
957    }
958
959    /**
960     * Create exception
961     */
962    public function createException($data)
963    {
964        // ...
965    }
966
967    /**
968     * Create exception
969     */
970    public function createException($data)
971    {
972        // ...
973    }
974
975    /**
976     * Create exception
977     */
978    public function createException($data)
979    {
980        // ...
981    }
982
983    /**
984     * Create exception
985     */
986    public function createException($data)
987    {
988        // ...
989    }
990
991    /**
992     * Create exception
993     */
994    public function createException($data)
995    {
996        // ...
997    }
998
999    /**
1000     * Create exception
1001     */
1002    public function createException($data)
1003    {
1004        // ...
1005    }
1006
1007    /**
1008     * Create exception
1009     */
1010    public function createException($data)
1011    {
1012        // ...
1013    }
1014
1015    /**
1016     * Create exception
1017     */
1018    public function createException($data)
1019    {
1020        // ...
1021    }
1022
1023    /**
1024     * Create exception
1025     */
1026    public function createException($data)
1027    {
1028        // ...
1029    }
1030
1031    /**
1032     * Create exception
1033     */
1034    public function createException($data)
1035    {
1036        // ...
1037    }
1038
1039    /**
1040     * Create exception
1041     */
1042    public function createException($data)
1043    {
1044        // ...
1045    }
1046
1047    /**
1048     * Create exception
1049     */
1050    public function createException($data)
1051    {
1052        // ...
1053    }
1054
1055    /**
1056     * Create exception
1057     */
1058    public function createException($data)
1059    {
1060        // ...
1061    }
1062
1063    /**
1064     * Create exception
1065     */
1066    public function createException($data)
1067    {
1068        // ...
1069    }
1070
1071    /**
1072     * Create exception
1073     */
1074    public function createException($data)
1075    {
1076        // ...
1077    }
1078
1079    /**
1080     * Create exception
1081     */
1082    public function createException($data)
1083    {
1084        // ...
1085    }
1086
1087    /**
1088     * Create exception
1089     */
1090    public function createException($data)
1091    {
1092        // ...
1093    }
1094
1095    /**
1096     * Create exception
1097     */
1098    public function createException($data)
1099    {
1100        // ...
1101    }
1102
1103    /**
1104     * Create exception
1105     */
1106    public function createException($data)
1107    {
1108        // ...
1109    }
1110
1111    /**
1112     * Create exception
1113     */
1114    public function createException($data)
1115    {
1116        // ...
1117    }
1118
1119    /**
1120     * Create exception
1121     */
1122    public function createException($data)
1123    {
1124        // ...
1125    }
1126
1127    /**
1128     * Create exception
1129     */
1130    public function createException($data)
1131    {
1132        // ...
1133    }
1134
1135    /**
1136     * Create exception
1137     */
1138    public function createException($data)
1139    {
1140        // ...
1141    }
1142
1143    /**
1144     * Create exception
1145     */
1146    public function createException($data)
1147    {
1148        // ...
1149    }
1150
1151    /**
1152     * Create exception
1153     */
1154    public function createException($data)
1155    {
1156        // ...
1157    }
1158
1159    /**
1160     * Create exception
1161     */
1162    public function createException($data)
1163    {
1164        // ...
1165    }
1166
1167    /**
1168     * Create exception
1169     */
1170    public function createException($data)
1171    {
1172        // ...
1173    }
1174
1175    /**
1176     * Create exception
1177     */
1178    public function createException($data)
1179    {
1180        // ...
1181    }
1182
1183    /**
1184     * Create exception
1185     */
1186    public function createException($data)
1187    {
1188        // ...
1189    }
1190
1191    /**
1192     * Create exception
1193     */
1194    public function createException($data)
1195    {
1196        // ...
1197    }
1198
1199    /**
1200     * Create exception
1201     */
1202    public function createException($data)
1203    {
1204        // ...
1205    }
1206
1207    /**
1208     * Create exception
1209     */
1210    public function createException($data)
1211    {
1212        // ...
1213    }
1214
1215    /**
1216     * Create exception
1217     */
1218    public function createException($data)
1219    {
1220        // ...
1221    }
1222
1223    /**
1224     * Create exception
1225     */
1226    public function createException($data)
1227    {
1228        // ...
1229    }
1230
1231    /**
1232     * Create exception
1233     */
1234    public function createException($data)
1235    {
1236        // ...
1237    }
1238
1239    /**
1240     * Create exception
1241     */
1242    public function createException($data)
1243    {
1244        // ...
1245    }
1246
1247    /**
1248     * Create exception
1249     */
1250    public function createException($data)
1251    {
1252        // ...
1253    }
1254
1255    /**
1256     * Create exception
1257     */
1258    public function createException($data)
1259    {
1260        // ...
1261    }
1262
1263    /**
1264     * Create exception
1265     */
1266    public function createException($data)
1267    {
1268        // ...
1269    }
1270
1271    /**
1272     * Create exception
1273     */
1274    public function createException($data)
1275    {
1276        // ...
1277    }
1278
1279    /**
1280     * Create exception
1281     */
1282    public function createException($data)
1283    {
1284        // ...
1285    }
1286
1287    /**
1288     * Create exception
1289     */
1290    public function createException($data)
1291    {
1292        // ...
1293    }
1294
1295    /**
1296     * Create exception
1297     */
1298    public function createException($data)
1299    {
1300        // ...
1301    }
1302
1303    /**
1304     * Create exception
1305     */
1306    public function createException($data)
1307    {
1308        // ...
1309    }
1310
1311    /**
1312     * Create exception
1313     */
1314    public function createException($data)
1315    {
1316        // ...
1317    }
1318
1319    /**
1320     * Create exception
1321     */
1322    public function createException($data)
1323    {
1324        // ...
1325    }
1326
1327    /**
1328     * Create exception
1329     */
1330    public function createException($data)
1331    {
1332        // ...
1333    }
1334
1335    /**
1336     * Create exception
1337     */
1338    public function createException($data)
1339    {
1340        // ...
1341    }
1342
1343    /**
1344     * Create exception
1345     */
1346    public function createException($data)
1347    {
1348        // ...
1349    }
1350
1351    /**
1352     * Create exception
1353     */
1354    public function createException($data)
1355    {
1356        // ...
1357    }
1358
1359    /**
1360     * Create exception
1361     */
1362    public function createException($data)
1363    {
1364        // ...
1365    }
1366
1367    /**
1368     * Create exception
1369     */
1370    public function createException($data)
1371    {
1372        // ...
1373    }
1374
1375    /**
1376     * Create exception
1377     */
1378    public function createException($data)
1379    {
1380        // ...
1381    }
1382
1383    /**
1384     * Create exception
1385     */
1386    public function createException($data)
1387    {
1388        // ...
1389    }
1390
1391    /**
1392     * Create exception
1393     */
1394    public function createException($data)
1395    {
1396        // ...
1397    }
1398
1399    /**
1400     * Create exception
1401     */
1402    public function createException($data)
1403    {
1404        // ...
1405    }
1406
1407    /**
1408     * Create exception
1409     */
1410    public function createException($data)
1411    {
1412        // ...
1413    }
1414
1415    /**
1416     * Create exception
1417     */
1418    public function createException($data)
1419    {
1420        // ...
1421    }
1422
1423    /**
1424     * Create exception
1425     */
1426    public function createException($data)
1427    {
1428        // ...
1429    }
1430
1431    /**
1432     * Create exception
1433     */
1434    public function createException($data)
1435    {
1436        // ...
1437    }
1438
1439    /**
1440     * Create exception
1441     */
1442    public function createException($data)
1443    {
1444        // ...
1445    }
1446
1447    /**
1448     * Create exception
1449     */
1450    public function createException($data)
1451    {
1452        // ...
1453    }
1454
1455    /**
1456     * Create exception
1457     */
1458    public function createException($data)
1459    {
1460        // ...
1461    }
1462
1463    /**
1464     * Create exception
1465     */
1466    public function createException($data)
1467    {
1468        // ...
1469    }
1470
1471    /**
1472     * Create exception
1473     */
1474    public function createException($data)
1475    {
1476        // ...
1477    }
1478
1479    /**
1480     * Create exception
1481     */
1482    public function createException($data)
1483    {
1484        // ...
1485    }
1486
1487    /**
1488     * Create exception
1489     */
1490    public function createException($data)
1491    {
1492        // ...
1493    }
1494
1495    /**
1496     * Create exception
1497     */
1498    public function createException($data)
1499    {
1500        // ...
1501    }
1502
1503    /**
1504     * Create exception
1505     */
1506    public function createException($data)
1507    {
1508        // ...
1509    }
1510
1511    /**
1512     * Create exception
1513     */
1514    public function createException($data)
1515    {
1516        // ...
1517    }
1518
1519    /**
1520     * Create exception
1521     */
1522    public function createException($data)
1523    {
1524        // ...
1525    }
1526
1527    /**
1528     * Create exception
1529     */
1530    public function createException($data)
1531    {
1532        // ...
1533    }
1534
1535    /**
1536     * Create exception
1537     */
1538    public function createException($data)
1539    {
1540        // ...
1541    }
1542
1543    /**
1544     * Create exception
1545     */
1546    public function createException($data)
1547    {
1548        // ...
1549    }
1550
1551    /**
1552     * Create exception
1553     */
1554    public function createException($data)
1555    {
1556        // ...
1557    }
1558
1559    /**
1560     * Create exception
1561     */
1562    public function createException($data)
1563    {
1564        // ...
1565    }
1566
1567    /**
1568     * Create exception
1569     */
1570    public function createException($data)
1571    {
1572        // ...
1573    }
1574
1575    /**
1576     * Create exception
1577     */
1578    public function createException($data)
1579    {
1580        // ...
1581    }
1582
1583    /**
1584     * Create exception
1585     */
1586    public function createException($data)
1587    {
1588        // ...
1589    }
1590
1591    /**
1592     * Create exception
1593     */
1594    public function createException($data)
1595    {
1596        // ...
1597    }
1598
1599    /**
1600     * Create exception
1601     */
1602    public function createException($data)
1603    {
1604        // ...
1605    }
1606
1607    /**
1608     * Create exception
1609     */
1610    public function createException($data)
1611    {
1612        // ...
1613    }
1614
1615    /**
1616     * Create exception
1617     */
1618    public function createException($data)
1619    {
1620        // ...
1621    }
1622
1623    /**
1624     * Create exception
1625     */
1626    public function createException($data)
1627    {
1628        // ...
1629    }
1630
1631    /**
1632     * Create exception
1633     */
1634    public function createException($data)
1635    {
1636        // ...
1637    }
1638
1639    /**
1640     * Create exception
1641     */
1642    public function createException($data)
1643    {
1644        // ...
1645    }
1646
1647    /**
1648     * Create exception
1649     */
1650    public function createException($data)
1651    {
1652        // ...
1653    }
1654
1655    /**
1656     * Create exception
1657     */
1658    public function createException($data)
1659    {
1660        // ...
1661    }
1662
1663    /**
1664     * Create exception
1665     */
1666    public function createException($data)
1667    {
1668        // ...
1669    }
1670
1671    /**
1672     * Create exception
1673     */
1674    public function createException($data)
1675    {
1676        // ...
1677    }
1678
1679    /**
1680     * Create exception
1681     */
1682    public function createException($data)
1683    {
1684        // ...
1685    }
1686
1687    /**
1688     * Create exception
1689     */
1690    public function createException($data)
```



8. MODELO DE CASOS DE USO

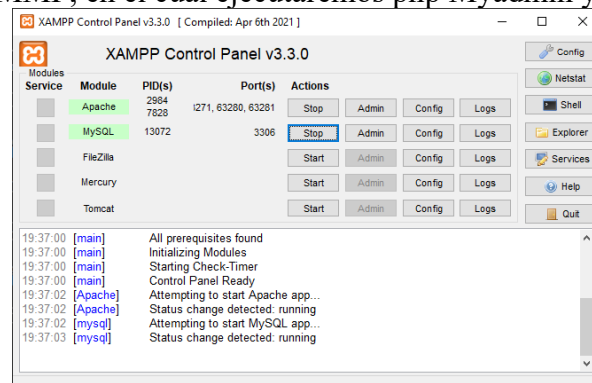


V2



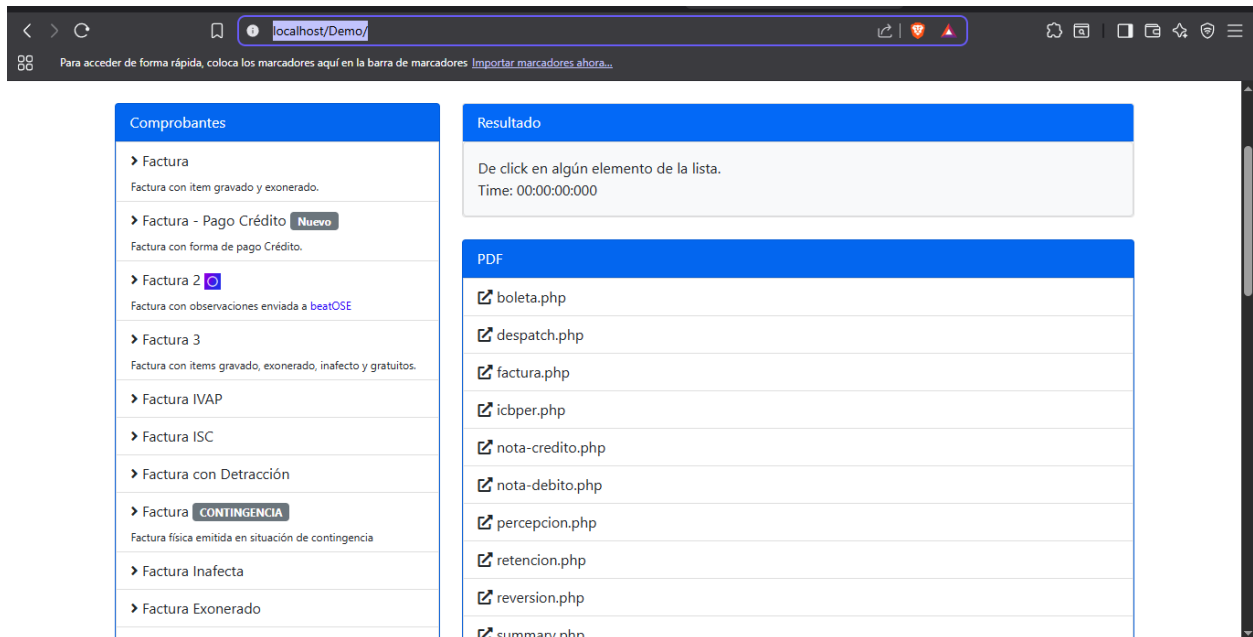
Análisis dinámico

Para el análisis dinámico con los requisitos configurados anteriormente abrimos nuestro servidor en este caso XAMPP, en el cual ejecutaremos php Myadmin y apache

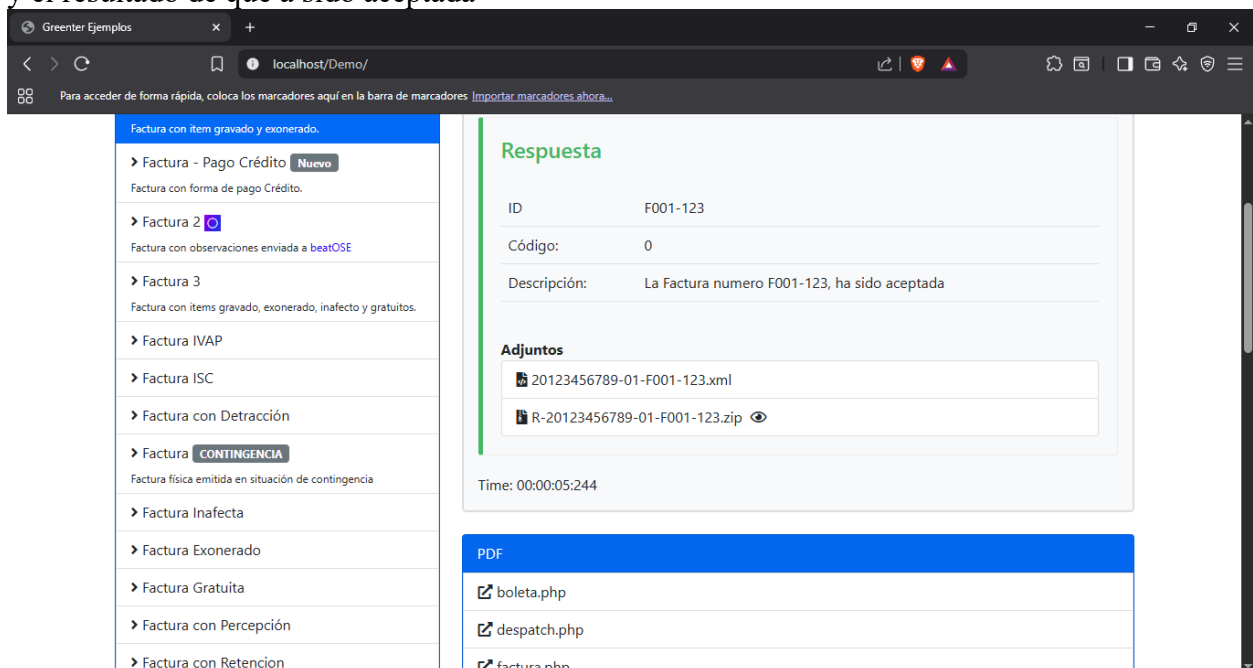


Una vez inicializados los servidores podemos ingresar a la página que está ubicada en <http://localhost/Demo/>.

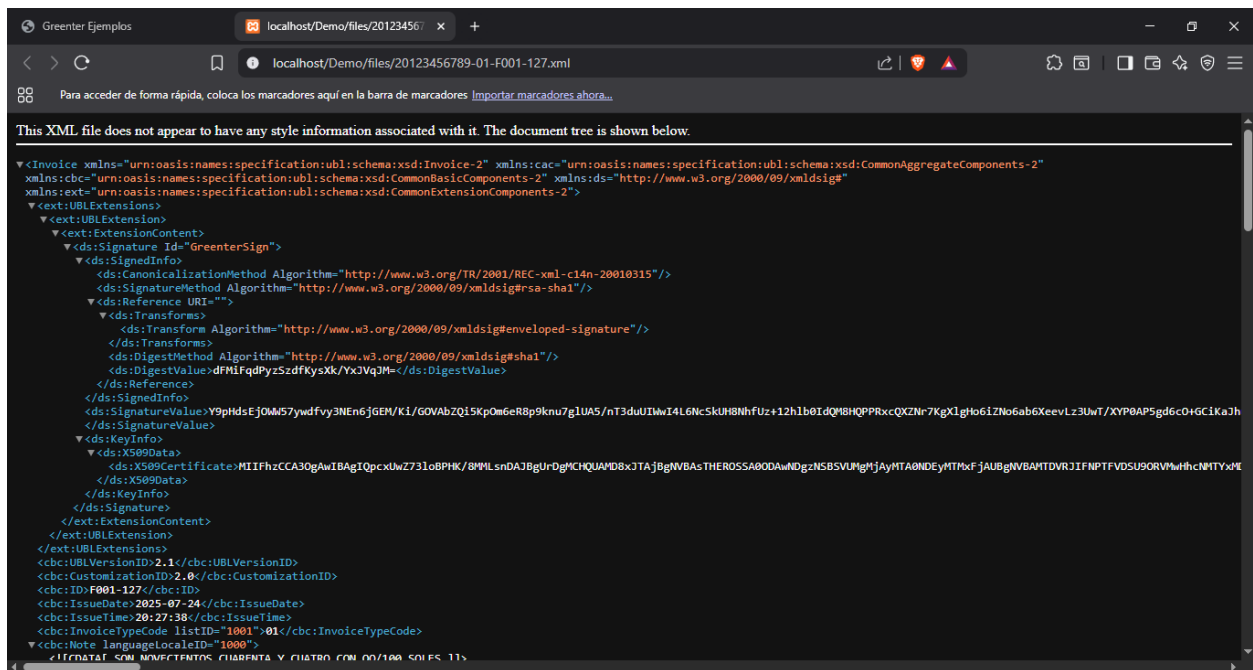
En el cual nos muestra lo que es un modo demo de archivos ya preconfigurados el cual nos muestra en su interfaz cómo funciona el sistema de facturación utilizando Sunat y greender en el cual nos muestra en el menú izquierdo diferentes comprobantes los cuales están listo para ser certificados mediante facturación electrónica.



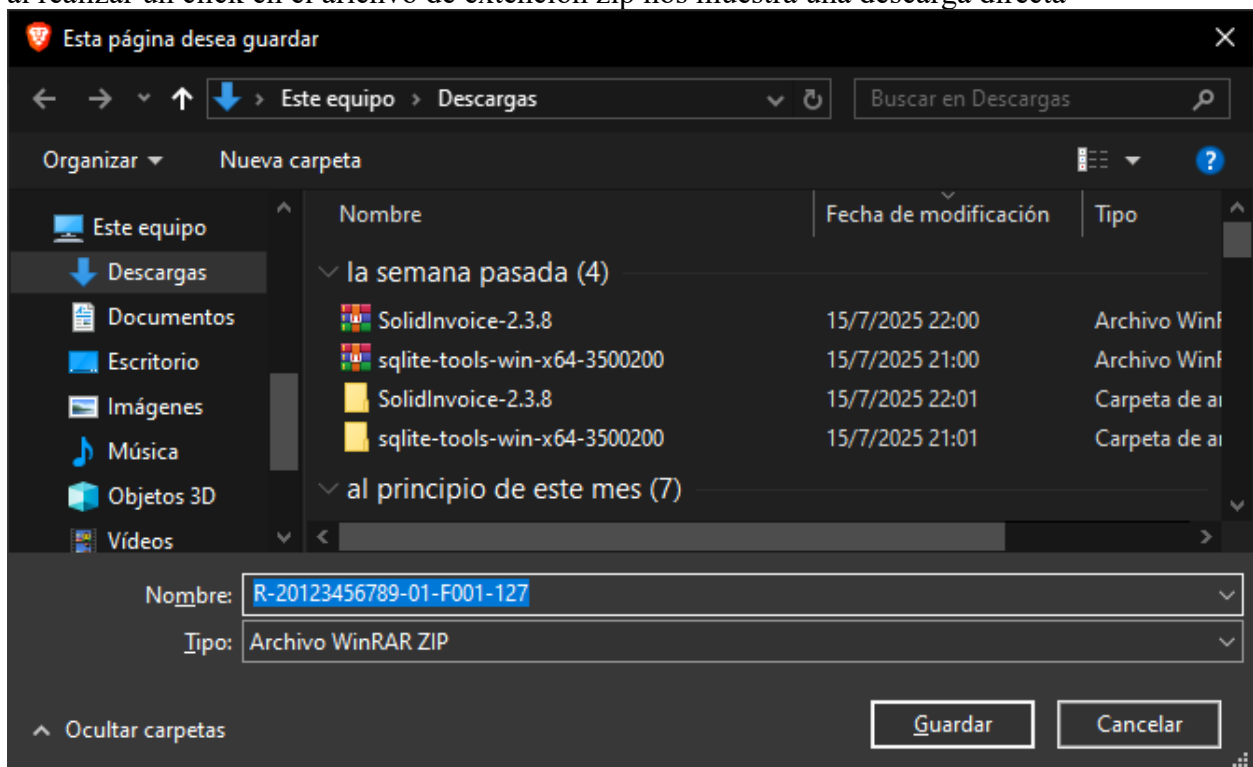
Al hacer click en alguna de las opciones de el menú lateral izquierdo se nos presenta en el apartado de resultado una opción en el cual nos da el tiempo de respuesta de del servidor desde el momento en el cual es enviado la plantilla xml para ser certificado por la api de SUNAT que en este caso es un tiempo de 5 segundos con 244 sesentesimas también nos muestra un conjunto de archivos adjuntos en formato xml y zip nos muestra el id de la factura el código y al descripción y el resultado de que a sido aceptada



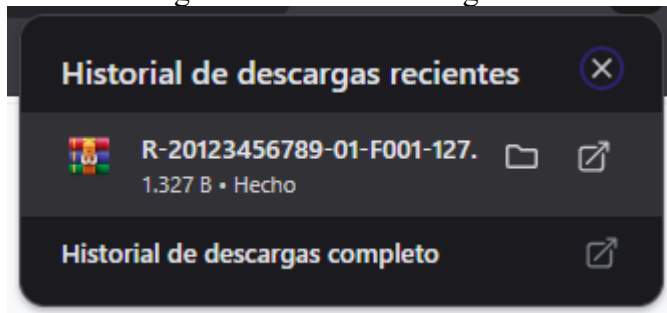
Si hacemos click en el archivo Xml no muestra como resultado la siguiente ventana, en el cual nos muestra los datos de la factura tambien datos de la certificacion y el ky de validacion con el cual a sido aceptada la fatura ademas de certificados como alf irma electronica de el ente que emite la factura en este caso como es de prueba el key que esta detallado en las imágenes de el analisis estatico, entre otros



al realizar un click en el archivo de extension zip nos muestra una descarga directa



De nos descarga un archivo ocn el sigueitne nombre



Al abrir la carpeta nos muestra los siguientes archivos los cuales son un archivo vacío llamado dummy de prueba podemos deducir y el archivo xml que verificamos al principio.

